ON DESIGNING USER INTERFACES FOR THE DESCRIPTION OF NETWORK SIMULATION EXPERIMENTS

by

William S. Stratton

A Proposal Submitted to the Honors Council

For Honors in the Department of Computer Science

September 22, 2014

Approved by: _____

L. Felipe Perrone
Thesis Advisor

_____

Brian R. King
Second Departmental Reader

_____

Stephen Guattery
Chair, Department of Computer Science

1 Introduction

The creation of a clear and logical interface is one of the most visible problems in the field of computer science. Various companies work toward this end and advertise their products as "easy" to understand for all types of users. This goes to show that not only the least technically literate can benefit from well designed interfaces. Even an expert using a new tool can appreciate a well designed interface that allows them to begin to work immediately and that prevents them from making mistakes. The user interfaces that cater to the needs of experts in computing do not always appear in the guise of the standard windows and buttons format that pervades modern life. They can also take the form of a specialized programming language created for that specific tool, which is often called a *domain specific language* (DSL).

The creation of languages is an activity spread throughout computer science with many notable examples, such as the well-known languages Java and Python. These languages have become popular because they are easy to read and to write and they are also expressive, that is, powerful enough to accomplish a variety of computational tasks. In contrast with general-purpose languages, DSLs are programming languages designed for one purpose. A field that can benefit from such languages is *network simulation*. Computer scientists and network engineers use simulation to study networks under situations where obtaining data and staging experiments is very difficult and/or very expensive. Simulation can help one to test new technologies, design new, more efficient communication protocols, and develop complex network infrastructure. In this setting, the user (either a novice or someone experienced) must communicate complex, structured information to a very specific type of program, the *simulator*.

The field of network simulation has faced a "crisis of credibility" [1, 2] primarily because published experiments were hard to replicate. The lack of reliable standards for describing experiments and the poor documentation of results are two factors that contributed to this crisis. In the last decade, experts have been working to resolve this issue in various ways. The

Simulation Automation Framework for Experiments (SAFE) [3] is one attempt to address this crisis by making it easy to produce repeatable, well documented experiments. SAFE achieves these goals by automating many aspects of the researcher's workflow. When SAFE is complete, anyone will be able to look at an experiment that has been published and see not just the results, but also the version of the simulator, the simulation settings, and any inputs. Replicating the experiment will be possible with minimal hassle.

We can identify two groups that will benefit from simulation automation systems like SAFE. The first is experts in network simulation who need full control over their experiments to harness the full power of simulation. We call such users *power users*; many of them prefer to work with a text-based interface known as a command-line interface. Such an interface is common throughout computer science for technical work that requires fine control over the system. The second group of users is those who are new to the field and wish to learn by replicating or honing experiments done by power users. We expect that most of these novice users will be students running and analyzing experiments set up by their professors. These users prefer a gentler, more intuitive interface that allows them to do everything they need and which protects them from errors.

In both of these cases, the user must submit information describing their experiment to the SAFE system in a format that it can understand. To this end, I have created the *SAFE Language for Experiment Description* (SLED). It currently has basic functions to allow a power user to define the configuration of an experiment, but it doesn't addresses all their needs. The main challenge in designing such a language is that it must be easy for power users to write and read, while at the same time being easy to generate automatically by the interface for novice users.

My thesis is that SAFE's ability to create credible and reproducible experiments will be enhanced by refinements to SLED that allow it to address the needs of both novice and power users.

2 Background

Simulation has an established process for experiment description. [4] One first creates an *experiment description*, which is a collection of pieces of information that completely specify the set of simulations that will be used to shed light on some investigative. This is where the independent variables are specified and a value is given to any variables that are held constant for the experiment. These independent variables are known as *factors*. Any factor has one or more values that will be used throughout the experiment. These values are known as *levels* and can vary across different runs of the simulator in one experiment. The cartesian product of the sets of levels associated with all factors creates a set all possible combinations of levels. These combinations are known as experimental *design points*, and the simulator is run at least once for each design point. The collection of design points in an experiment is known as the *Design Of Experiment* (DOE) space. Formalisms or languages to describe the DOE space already exist. In the remainder of this section, I introduce some of the languages I am investigating to drive the development of SLED.

The *Scripts for Organizing 'Speriments* (SOS) [5] defines a very simple language. From the information provided by the user in a text file, it uses combinatorics to create a full list of design points and then prunes down that list based on the specified restrictions. These concepts were later adapted by the predecessor to SAFE, a system called *SWAN Tools*. [6] Although SOS and SWAN Tools are longer active projects, they were valuable first steps down the road to well documented, automated simulation. From its inception, SAFE has followed the path that these projects defined, even in its first attempt at an experiment description language.

The *ns-3 Experiment Description Language* (NEDL) [7] was originally designed for the needs

of the SAFE project. NEDL was based on the *eXtended Markup Language* (XML) which is used to organize and to label text-based data. This was a good choice at the time for two reasons. First, the structure of XML is similar to that of the language used to design webpages, allowing anyone with web design experience to be able to understand and to write NEDL easily. Second, because there exist many freely available tools for parsing XML, instrumenting SAFE to process NEDL was very simple. However, NEDL had shortcomings, as well. XML is a very verbose language, so both reading and writing it take some time. As originally conceived, NEDL was ambitious and the language's structures addressed the needs of SAFE, but the language was hard to work with. The XML foundation allowed the use of tools for parsing, but there are other languages that can do the same job more cleanly and compactly.

SAFE's requirement of supporting novice users inspired the design of a *web-based user interface* (WUI) for creating experiments. The WUI is basically a carefully designed website and my plan is to use JavaScript in its implementation. This will allow me to keep a friendly look and feel and to interconnect easily with the WUI easily with the SAFE experiment execution manager. The information entered by the novice user can be encoded in a format known as *JavaScript Object Notation* (JSON) [8] and then sent to SAFE. The experiment's information encoded in JSON is saved in a database and parsed by SAFE to execute the experiment. In the course of my summer research on this process, it became clear that the webpage→JSON→server data flow was very similar to the existing procedure on the server to process XML from user input through a command-line interface. I realized then that the programs on the server can be streamlined if they only have to interpret one type of description language, so I decided to consolidate the two processes into one. Figure 1 illustrates how I propose to change the processing of experiment descriptions in SAFE. This new process will allow the server to deal with only one type of file, regardless of which interface generates it (command-line or WUI).

As an XML-based language, NEDL has a number of interesting features that would be interesting to keep in the JSON-based new language SLED. My work will explore this approach

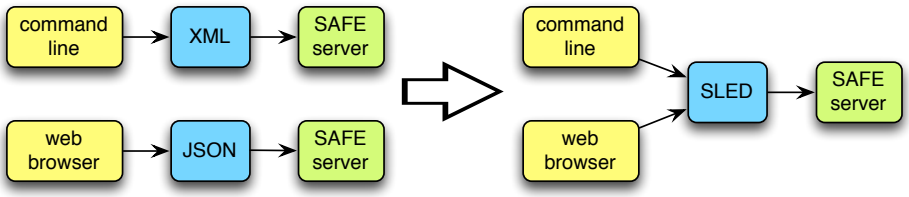

Figure 1: Modifying the processing of experiment descriptions in SAFE.

and, hopefully, make the users' lives easier and SAFE's processing of experiment descriptions more efficient. I expect that SLED will include the same language features of NEDL, but will be easier to read and to write by power users, simple to generate automatically by the WUI for novice users, and faster to process in the SAFE server. By the end of this project, I expect that SLED will fully replace NEDL in SAFE and more importantly, that it will have addressed the needs of readability, automatic generation, and expression of the experiment's needs.

3 Project Description

The first major task in developing SLED to its full potential is researching other simulation languages and their capabilities. This will give me material to think critically about the language features I have already defined and which might still need to be added. I have identified three similar languages to study in depth. The first is for a popular network simulator called OMNet++. [9] This language requires intimate knowledge of the code that will be run on the simulator, which makes it harder to use than SLED. However, it includes features for control over individual experiment runs and other aspects of the simulation that SLED does not need. The second is NEDL [7], as it reflected the first plan of what SAFE needs in an experiment description language. The third language is for general purpose simulation and called *Simulation Experiment Specification via a Scala Layer* (SESSL). [10] The ambition of the project means that it has far more features than those needed in SAFE. As far as I have been able to determine, it also provides limited capabilities for integration with web pages. However, since SESSL

implements a variety of features while maintaining readability, it will serve as a useful model in designing the more advanced features of SLED.

My preliminary research into these languages has allowed me to identify two improvements to SLED. First, I will implement a feature known in Python as *list comprehension*, so that factors do not need to be enumerated sequentially as in {`1,3,5,7,9`}. A notation like {`1 to 9 by 2`} is more compact and easier to read than the several lines of XML code required by NEDL. Second, I will implement a *validation* feature on factors to restrict the levels that can be assigned to them. For example, a factor to express the "speed of a mobile device" would be given a set of constraints like {`min: 0m/s, max: 10m/s`} so that a novice user would be protected from entering levels that would configure a meaningless experiment.

After creating these pieces and any others that I discover during my research, I will work on designing a web site (or WUI) to guide users in creating experiments. My focus will be more on the functionality than on the look and feel of this web site; the visual aspects can be refined by someone else once the structure is in place. This will create a three-tiered system for user interaction with SAFE. At the top is there will be the web site, where the novice user can create experiments one (the work I will do for this project) and also see graphics made from experimental results (the completed work of another student's summer research project). The web site will let the novice user enter data through standard forms and will automatically generate a SLED file. The power user can bypass the web site and write the SLED files manually. Once that information on the experiment's configuration is gathered, it will be sent to the main SAFE system for processing and execution. Figure 2 shows this structure.

The main code of SAFE is written in the *Python* programming language and makes heavy use of a web application framework called *Django*, which manages the interactions with a database that stores the experiments' descriptions and results. The preliminary code I have already written to implement a foundation for SLED processing makes use of both of these technologies. I will continue to use these programming tools and strive to shape the system into a cohesive whole.

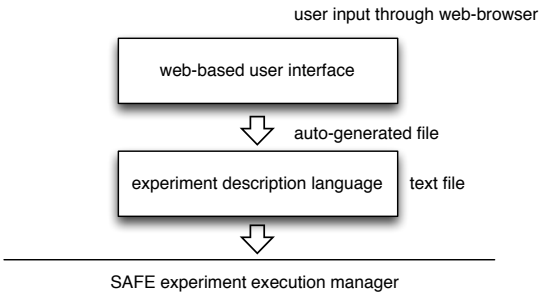

Figure 2: Modifying the processing of experiment descriptions in SAFE.

On the web site, I will use JavaScript to support the user interactions, to generate SLED files, and submit them to the SAFE backend. JavaScript will enable me to create a responsive interface that makes for a smooth user experience.

4 Conclusion

A user interface that is intuitive and yet powerful will enable SAFE to serve a broad audience, including students who would not otherwise have the opportunity to get hands-on experience with network simulation. By further developing SLED and integrating it with a user interface for novices, I will bring to SAFE the balance of the needs of two user profiles. The work that I will do lies mainly in the fields of programming language design and web-based applications. When finished, this work will reduce significantly a barrier to entry in the field of network simulation. This means easier access to users who can benefit from the use of the technology but don't have the background to apply it. My work will also help advanced users to get what they need from the SAFE system through the command line interface that appeals more directly to them. The completion of SLED and SAFE user interfaces will be the culmination of a four year long project. The advances I will make to the SAFE project will push it closer to a state where it can be publicly released, so I will be making a contribution to those who work on improving existing and future networks.

References

[1] K. Pawlikowski, H.-D. J. Jeong, and J.-S. R. Lee, "On credibility of simulation studies of telecommunication networks," IEEE Communications Magazine, **40**, 132 (2002).

[2] K. Pawlikowski, "Do not trust all simulation studies of telecommunication networks," in "International Conference on Information Networking (ICOIN 2003)," (Jeju Island, Korea, 2003).

[3] L. F. Perrone, C. S. Main, and B. C. Ward, "SAFE: Simulation Automation Framework for Experiments," in C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds., "Proceedings of the 2012 Winter Simulation Conference," (IEEE, Piscataway, NJ, 2012).

[4] A. Law, *Simulation Modeling and Analysis*, 4th ed. (McGraw-Hill, New York, 2007).

[5] T. G. Griffin, S. Petrovic, A. Poplawski, and B. J. Premore, "SOS: Scripts for Organizing 'Speriments," (2002), available at `http://www.ssfnet.org/sos/README`. [Accessed Sept. 15, 2014].

[6] L. F. Perrone, C. J. Kenna, and B. C. Ward, "Enhancing the credibility of wireless network simulations with experiment automation," in "Proceedings of the 2008 IEEE International Conference on Wireless & Mobile Computing, Networking & Communication (WiMob '08)," pp. 631–637 (IEEE Computer Society, Washington, DC, USA, 2008).

[7] A. W. Hallagan, "The design of XML-based model and experiment description languages for network simulation," Honors Thesis, Bucknell University, 2010.

[8] E. International, "Introducing JSON," available at `http://www.json.org` [Accessed Sept. 20, 2014].

[9] OMNeT++, "OMNeT++ user manual - 9 configuring simulations," available at `http://www.omnetpp.org/doc/omnetpp/manual/usman.html#sec331` [Accessed Sept. 20, 2014].

[10] R. Ewald and A. M. Uhrmacher, "SESSL: A domain-specific language for simulation experiments," ACM Transactions on Modeling and Computer Simulation, **24 2** (2014).